Занимательный компьютер

О вирусах, **червяках и войне в памяти**

А.К. ДЬЮДНИ

«По-настоящему безопасной можно считать лишь систему, которая выключена, замурована в бетонный корпус, заперта в помещении со свинцовыми стенами и охраняется вооруженным караулом, - но и в этом случае сомнения не оставляют меня».

Юджин X. Спаффорд

В ДВЕРЬ постучали. По всему чувствовал ось, что произошло что-то чрезвычаЙное.Начальник вычислительного центра оторвал голову от разложенных на рабочем столе бумаг. В кабинет вошел оператор машинного зала. «Что-то случилось. Творятся непонятные вещи. У машины не хватает памяти. Наверное, в систему проник вирус». Если бы центр был оборудован аварийными сиренами, директор наверняка включил бы их.

Подобная сцена (с теми или иными вариациями) была нередким явлением в последние годы. Вслед за этим в выпусках новостей все чаще появлялись сообщения о новых жертвах компьютерных вирусов. В них неоднократно упоминал ось и о статьях из рубрики «Занимательный компьютер» в связи с разрастающейся вирусной эпидемией среди компьютеров, возможно, потому, что мы неоднократно публиковали статьи об игре «Война в памяти» - о компьютерных программах, уничтожающих друг друга. Однако, как мы увидим, «Война в памяти» не имеет прямо го отношения к вирусам, поражающим компьютеры.

Чтобы понять, как действует такой вирус, необходимо детально знать систему, в которой он обитает. То же самое можно сказать и о принципах, лежащих в основе действия «червяков», «логических бомб» и других поражающих средств, способных вывести из строя компьютер. В этой связи следует указать на два обстоятельства. Во-первых, журналисты в своих сообщениях склонны неправильно освещать, а порой и вовсе искажать истории о вирусах, причем неумышленно: большинство репортеров имеют довольно смутное представление о внутреннем устройстве и работе компьютера. Во-вторых, описанием компьютерного вируса в журнальных статьях - даже относительно подробном - никто не сможет воспользоваться для его практического создания, за исключением тех, кто обладает всеми необходимыми познаниями о компьютерной системе, в которой должен оперировать вирус. Однако знающий «технопат», помешавшийся на идее уничтожения программ и данных в том или ином компьютере, едва ли нуждается в журнальных или газетных статьях, чтобы придумать способ создания вируса. Поэтому нет никакого вреда в том, чтобы описать, каким образом действует вирус и другие разрушительные программы. (На самом деле, такое описание, возможно, даже окажется полезным, если сыграет роль дополнительного стимула в усилиях, направленных на защиту компьютерных систем.)

Следует с самого начала провести различие между двумя наиболее распространенными типами вредных программ. «Вирус», если можно так выразиться, едет верхом на другой программе, он не может существовать самостоятельно. «Червяк» же способен на независимое существование, и его поведение в чем-то схоже с поведением бактерии. Оба вида «инфекции», как и всякая программа, зависят от операционной системы.

Большинство читателей, наверное, знают, что всякий работающий компьютер включает в себя материальную часть (аппаратные средства) и программное обеспечение. Например, сейчас передо мной компьютер Apple IIc, а в его памяти находится программное обеспечение - программа для обработки текстов «Appleworks Word Processor». Эта программа переводит буквы, которые я набираю на клавиатуре компьютера, в ту область памяти машины, которая отведена программой для текста.

Однако программа обработки текстов не может работать сама по себе. Ей необходима операционная система, которая, помимо прочих функций, переводит программу на специальный машинный язык, позволяющий аппаратным средствам машины выполнять ее инструкции. Операционная система персонального компьютера обычно находится на магнитном диске. Чтобы начать работать с такой машиной (скажем, написать какой-нибудь текст или поиграть в игру), необходимо предварительно загрузить дисковую операционную систему (ДОС) в электронную память компьютера. В домашнем компьютере ДОС обычно автоматически загружается с диска, который может содержать или не содержать нужную про грамму , как только включается питание.

Чтобы выполнить нужную программу на моем персональном компьютере, я должен сначала набрать ее имя на клавиатуре. После этого ДОС отыщет на диске программу с этим именем, загрузит ее в память и потом будет выполнять одну инструкцию за другой, как показано на среднем рисунке на с. 83.

При загрузке программы ДОС отводит в электронной памяти место не только для самой программы, но и для ее «рабочего пространства». Здесь программа будет хранить все значения, присваиваемые ее переменным, массивам и т. п. При этом ДОС принимает специальные меры предосторожности, чтобы не уничтожить другие программы или структуры данных, включая ту часть операционной системы, которая в данный момент находится в памяти. Аналогичные меры предосторожности ДОС принимает при записи программ или данных на диск.

Нередко про грамм ист считает, что ему необходимо использовать команды самой операционной системы. Эти команды обычно можно найти в руководствах по ДОС. С помощью этих команд можно написать подпрограмму, которая читает файлы с диска и помещает их в оперативную память, изменяет файлы и записывает их обратно на диск - иногда с недобрыми намерениями.

Приведем пример подпрограммы-вируса, действующей по такому принципу. Она состоит из псевдокоманд ДОС и подпрограмм: маленьких внутренних программ (составляющие их инструкции хранятся отдельно от главной программы), выполняющих некоторые специальные функции всякий раз, когда к ним обращаются.

```
this := findfi1e
LOAD (this)
loc := search (this)
insert (loc)
STORE (this)
```

Подпрограмма под названием findfile обращается к каталогу выполняемых файлов, или программ, на диске, берет произвольное имя файла и присваивает имя этого файла переменной *this* (этот). В следующей строке программы используется псевдокоманда ДОС LOAD (загрузить), с помощью которой файл помещается в оперативную память компьютера. Другая подпрограмма под названием search (поиск) просматривает только что загруженную программу в поисках инструкции, которая могла бы послужить подходящим местом, когда можно занести вирус. Когда процедура search находит такую инструкцию, она определяет соответствующий номер строки и присваивает его в качестве значения переменной *loc*.

Теперь все готово для того, чтобы подпрограмма-вирус могла проникнуть в произвольно выбранную из каталога программу. Подпрограмма insert (вставить) заменяет выбранную инструкцию другой (например такой, как вызов подпрограммы), которая передает управление блоку команд, составляющих главное тело подпрограммы-вируса, присоединяющейся к концу программы. Затем к концу добавленной подпрограммы присоединяется инструкция, возвращающая управление «зараженной» программе, на инструкцию, следующую за вставленной.

Таким образом, когда выполняется подпрограмма-вирус, выполняется также и подмененная инструкция заряженной программы. Исходная программа работает так, будто ничего особенного не произошло. Однако, на самом деле, подпрограмма-вирус воспользовалась каким-то мгновением, чтобы захватить власть над средствами операционной системы и прицепить свою копию к еще одной программе, хранящейся на диске. Этот процесс в графической форме представлен правой частью рисунка внизу. Когда зараженная таким образом программа будет затем загружена операционной системой в память компьютера, то в процессе выполнения она заразит еще одну программу на диске, хотя внешне ее работа будет выглядеть вполне нормально.

Еще в 1984 г. Фред С. Койен провел управляемый эксперимент по заражению в Университете Южной Калифорнии и выяснил, к своему немалому удивлению, что вирусы, аналогичные тому, который был только что описан, могут заразить целую компьютерную сеть всего за каких-нибудь несколько минут. Чтобы дать читателю представление о том, какой вред способны принести подобного рода вирусы, я несколько адаптирую исходный вирус Койена, записав его на псевдоалгоритмическом языке:

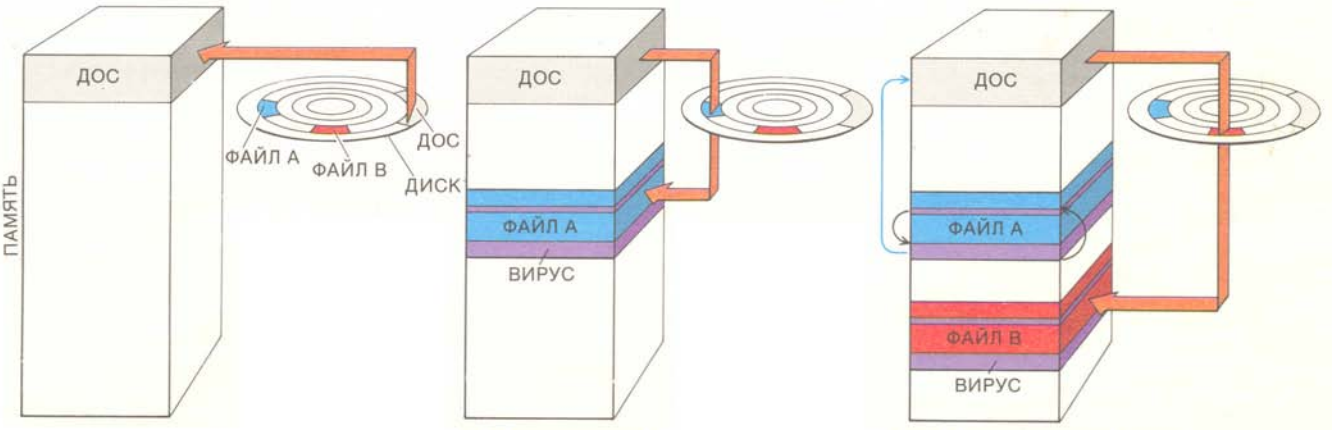
```
1234567
главная программа:
1. infect
2. if условие удовлетворяется then навредить
3. go to исходная программа
подпрограмма: infect
1. выбрать произвольно выполняемый файл
2. if первая строка файла = 1234567 then go to 1, else присоединить спереди вирус к файлу
подпрограмма: «условие удовлетворяется»
подпрограмма: «навредить»
```

Вирус Койена отличается местом присоединения: вместо того чтобы забираться в середину или конец исходной программы, он прикрепляется к ее началу. Первая строка программы вируса - это «узнавание кода» 1234567. Главная программа сначала вызывает подпрограмму infect (заразить), которая случайным образом выбирает выполняемый файл на диске и проверяет, начинается ли первая строка этого файла с 1234567. Если да, то данная программа уже была заражена, и подпрограмма infect выбирает какую-нибудь другую программу. Если находится незараженная программа, то перед ее началом вставляется вся программа-вирус. Это означает, что программа-вирус всегда будет выполняться в первую очередь и лишь затем возвращать управление зараженной программе.

Следующие две процедуры проверяют выполнение определенного условия и совершают определенное разрушительное действие. Условием срабатывания может быть, например, определенная дата, достигнутая системным часовым механизмом, или удаление фамилии какого-нибудь служащего из списка сотрудников, которым начисляется заработная плата. В качестве разрушительного действия может быть стирание всех файлов или более «тонкое» вредительство - замена нескольких битов в некоторых местах. Возможности тут безграничны.

Условия срабатывания и разрушительные действия уже подводят нас к границе, за которой начинаются аморальные поступки. Я думаю, что ни у кого не вызывает сомнения то, что причинение ущерба компьютерным файлам при помощи вируса или какой-нибудь другой разрушительной программы - это предосудительное действие. Компьютерные программы и данные - это не просто безликие цепочки нулей и единиц. В них вложена мысль человека. Зачем же разрушать то, что было сделано другими?

Создатели программ-вирусов делают это по разным причинам. Например программист , служащий в большой компании, может, затаив обиду



При загрузке ДОС в память (слева) *может быть прочитана программа* (в середине) *и вирус* (справа) *может создать свою копию*

КОМАНДА	ПОЯСНЕНИЯ
DAT B	Невыполнимая команда; B - это значение элемента данных
MOV A B	Переместить содержимое ячейки A в ячейку B
ADD A B	Сложить содержимое ячеек A и B
SUB A B	Вычесть содержимое ячейки A из содержимого ячейки B
JMP B	Передать управление на ячейку B
JMZ A B	Передать управление на ячейку A, если в ячейке B находится O
JMN A B	Передать управление на ячейку A, если в ячейке B находится величина, не равная нулю
DJN A B	Вычесть 1 из содержимого ячейки B и передать управление по адресу A, если содержимое B не равно O
CMP A B	Сравнить величины в ячейках A и B; если они равны, пропустить следующую команду
SPL B	Разветвить выполнение между следующей командой и командой по адресу B

Система команд для «Войны в памяти»

на правление фирмы, запустить в систему вирус, который срабатывает в день его увольнения из компании. Когда это случится, фамилия программиста исчезнет из списка сотрудников, которым начисляется зарплата, и это послужит вирусу сигналом. В результате ценные данные и программы, принадлежащие компании, будут либо уничтожены, либо в них появятся серьезные и дорогостоящие ошибки. Замена поврежденных данных и про грамм их запасными копиями, хранимыми на других носителях, может не всегда приводить к желаемому результату, потому что, если копии сделаны недавно, они тоже могут оказаться зараженными.

Конечно, разрушения, подобные описанным выше, могут произойти лишь в компьютерной системе коллективного пользования. Операционные системы таких компьютеров значительно сложнее, чем дисковая операционная система персонального компьютера. Во-вторых, уже тот факт, что одними и теми же ресурсами пользуется не один человек, а многие, означает, что операционная система должна как можно лучше защитить одних пользователей от случайного или преднамеренного вмешательства со стороны других. Даже в такой среде могут существовать вирусы, но создавать их уже значительно сложнее. Обычно они пользуются каким-либо слабым звеном в операционной системе - ошибкой или недосмотром, - как показывает пример «вируса» (на самом деле правильнее было бы назвать его «червяком»), поразившего прошлой осенью компьютерную сеть Internet.

Вечером 2 ноября 1988 г. кто-то запустил программу «червяк» с одного из нескольких тысяч компьютеров, соединенных в сеть передачи данных под названием Internet. Эта сеть объединяет машины университетских центров, частных фирм, правительственных агентов, таких как Национальное управление по аэронавтике и исследованию космического пространства, и даже некоторые военные учреждения. С ужасающей быстротой червяк распространился более чем на 1000 машин менее чем за два дня. По мере того как размножались копии червяка, операторы различных вычислительных центров стали замечать, как увеличиватся размер занятой памяти и замедляются ответы машины. Однако червяк не поражал файлы или другие программы. Казалось, его целью было лишь проникнуть как можно в большее число машин сети. Но даже при этом условии ущерб, выразившийся в количестве потерянного времени, был колоссальным.

Как уже указывал ось выше, червяк - это программа, способная самостоятельно существовать в компьютерной среде; она переходит из машины в машину, повсюду оставляя свои копии. Червяк, поразивший сеть Internet, состоял из двух частей: вектора и тела. Начав работать в каком-то компьютере, червяк ищет другую машину, засылая туда свой вектор. Оказавшись внутри машины, вектор устанавливает канал связи, по которому можно переслать тело червяка. Подробности этого нападения были раскрыты Юджином Х. Спаффордом из Университета Пардю в 40-страничном документе, опубликованном через несколько недель после происшествия. Об изобретательности автора программы-червяка свидетельствует следующий пример.

Операция система UNIX, предпочитаемая многими пользователями сети Internet, допускает в машине процессы, не связанные ни с одним конкретным пользователем. Такие независимые процессы называются «демонами». Один подобный демон fingerd (произносится как «фингерди») дает возможность пользователям получить информацию о других пользователях. Наличие такого сервиса очень желательно в компьютерной среде, когда многие пользуются одними и теми же про граммами и структурами данных, предназначенными для научных и технических расчетов.

Про грамма-червяк, работающая в данный момент в каком-то компьютере, посылает сообщение одному или нескольким другим компьютерам, потенциальным жертвам поражения, имеющимся в списке (полученном незаконным путем). Запросив услуги демона fingerd, червяк передает ему некоторую информацию, так же, как это делает любой другой нормальный пользователь. Однако червяк сообщает демону так много информации, что она заполняет всю область памяти, отведенную демону в компьютере, и проникает в «запретную зону».

Заполненная таким образом зона при нормальных условиях отводится под инструкции, выполняя которые fingerd решает, что следует делать дальше. Попав в эту зону, червяк (тело которого все еще находится в исходной машине), активирует так называемый интерпретатор команд новой машины, по существу завладев небольшой частью операционной системы UNIX. После того как командный интерпретатор поступил в его распоряжение, червяк передает 99 строк исходного кода, составляющего вектор. Ничего не подозревающая жертва, новый компьютер, компилирует и выполняет программу вектора, и теперь заражение уже фактически гарантировано.

Программа вектор прячется в системе, меняя свое имя и ликвидируя все файлы, созданные в процессе его проникновения в систему. После этого вектор устанавливает новый канал связи с исходной машиной и с помощью стандартных системных протоколов копирует файлы, составляющие основное тело червяка.

После того как червяк проник в новую машину, его главная задача заключается в том, чтобы раскрыть имена и адреса других машин, про-

никнув в области памяти, отведенные законным пользователям системы. Для этого червяк применяет хитроумную процедуру угадывания паролей, которая, благодаря небрежности большинства пользователей в выборе паролей, оказалась весьма эффективной. Раскрыв пароль законного пользователя, червяк «прикидывается» этим пользователем для того, чтобы прочитать то, что он записал в память компьютера, и раскрыть имена других компьютеров сети Internet, чтобы заразить их.

Как говорит Спаффорд, в большую часть средств операционной системы UNIX (охарактеризованных им как неудачные), позволивших червяку действовать так успешно, были внесены соответствующие коррективы. Однако этот факт не устранил его обеспокоенности в отношении компьютерной безопасности, как видно из цитаты, взятой в качестве эпиграфа к настоящей статье. Возможно, он думал о теоретическом исследовании вирусов, проведенном Койеном, которое может оказаться справедливым и для червяков.

Если технопаты проявят упорство в своих варварских нападениях на компьютерные системы, то, возможно, самое время организовать Центр по борьбе с вирусами. Во время нашествия червяков на сеть Internet специалисты Калифорнийского университета в Беркли и нескольких других центров, подключенных к сети, сумели «перехватить» несколько копий червяка, проанализировали его инструкции и установили, каким образом он работает. По-видимому имеет смысл учредить национальное агенство, которое боролось бы против вирусов и червяков, когда бы и где бы они не были обнаружены; такая служба будет необходимой, если компьютерные инфекции не престанут распространяться. Хотя случай с сетью Internet - это лишь намек на ужасы, к которым может привести нашествие вирусов, он также продемонстрировал, насколько эффективным может быть организованное сопротивление.

Койен установил, что невозможно написать компьютерную программу, способную обнаружить вирус любой природы, хотя можно найти противоядие против любого известного вируса. С другой стороны, на любое противоядие найдутся другие вирусы, которые смогут обойти его. По мнению Койена, такое угрожающее положение дел в будущем может породить компьютерные среды со своеобразным эволюционным процессом, в котором выживут лишь самые «сильные» программы.

Эта ситуация напоминает нам о «Войне в памяти», компьютерной игре, о которой уже говорилось в нескольких статьях в рубрике «Занимательный компьютер» (см. «В мире науки», 1984, № 7; 1985, № 5 и 1987, № 3). Однако «Война в памяти» не трогает ни в чем неповинных компьютерных систем. Ее программы ищут лишь себе подобных - другие программы, предназначенные для игры. Две программы вступают в бой в специально отведенной области компьютерной памяти, называемой колизеем. Не существует опасности, что программа игры «Война в памяти» когда-нибудь вырвется на свободу и причинит вред кому бы то ни было, потому что программа «Войны в памяти» или любая подобная ей программа никогда не сможет работать в нормальной компьютерной среде. Программы этой игры написаны на специальном языке «Редкод». Основные команды этого языка сведены в таблицу, приведенную на с. 84.

Простой пример такой программы поможет ввести в курс дела читателей, еще незнакомых с игрой. Ниже приведена программа DWARE (карлик), которая бросает нулевые бомбы в каждую пятую ячейку памяти:

```
DAT  -1
ADD  #  5  -1
MOV  #  0  @  -2
IMP  -2
```

Арена колизея (область памяти, в которой обитают все программы «Войны в памяти») состоит из нескольких тысяч адресов, или пронумерованных ячеек памяти, организованных в длинную непрерывную полосу. Инструкции, составляющие программу DWARF, например, занимают 4 последовательных ячейки колизея с адресами, скажем, 1001, 1002, 1003 и 1004.

ИнструкцияDAT служит для того, чтобы хранить число, используемое в программе (в данном случае -1), и записана в ячейку с адресом 1001. Инструкция ADD прибавляет число 5 к содержимому ячейки, отстоящей на -1 адрес от инструкции ADD. Поскольку последняя имеет адрес 1002, то число 5 прибавляется к числу, хранящемуся по предшествующему адресу, т. е. по адресу 1001, заменяя число -1 на 4. Инструкция MOV помещает число 0 в ячейку памяти с адресом определенным как @ -2. Где это? Этот адрес вычисляется путем обращения к инструкции DAT, расположенной на две ячейки впереди от инструкции MOV. Там находится адрес ячейки, куда программа должна поместить 0. Заключительная инструкция JMP передает управление программе DWARF на две инструкции назад, т. е. возвращает ее на инструкцию ADD. В результате весь процесс повторяется снова.

После вторичного выполнения, программа заменит содержимое ячейки с инструкцией DAT на 9 и поместит 0 в ячейку с этим адресом. Если по этому адресу хранится одна из инструкций программы противника, она будет стерта и, так как на ее месте окажется невыполнимая команда, программа противника может «погибнуть».

Программа DWARF бросает нулевые бомбы в каждую пятую ячейку памяти до тех пор, пока не достигнет последнего адреса - но арена нигде не кончается, потому что последний адрес прилегает к первому. Следовательно бомбы, разбрасываемые «карликом» начинают падать все ближе и ближе к его собственным ячейкам. Однако, поскольку длина «карлика» равна всего 4 ячейкам, а общее число ячеек памяти обычно кратно 10, программа избегает случайных попаданий своих собственных бомб, продолжая жить и сражаться - хотя вслепую и довольно глупо.

За последние несколько лет «Война в памяти» превратилась в весьма замысловатую игру с многочисленными и разнообразными стратегиями и контрстратегиями. Существуют программы, разбрасывающие повсюду свои копии, программы, стреляющие «безмозглыми» боевыми программами, состоящими лишь из одной инструкции, и даже такие программы, которые «ремонтируют» себя после вызванных атакой неприятеля разрушений.

Международная ассоциация любителей игры «Война в памяти» со штаб-квартирой в Лонг-Биче (шт. Калифорния) и филиалами в Италии, Японии, Польше, Советском Союзе и ФРГ ежегодно организует турниры, в которых программисты могут испытать свое искусство. Читатели, желающие вступить в ассоциацию любителей игры, могут написать Уильяму Р. Бакли по адресу: William R. Buckley, 5712 Kern Drive, Huntington Beach, Calif. 92649.

В турнире 1987 г. программы, присланные из Японии, задали немало хлопот боевым программам Северной Америки. А победителем последнего турнира, состоявшегося в декабре прошлого года в Лос-Анджелесе, оказалась программа из Советского Союза, получившая, как это ни странно, имя COWBOY (ковбой). Написанная Е. Лилитко из Переславля-Залесского, небольшого города недалеко от Москвы, программа COWBOY следила за «бомбометани-

ем» программ противника, уходя из опасной зоны и затем переходя к мощным контратакующим действиям. Лилитко выиграл первый приз в 250 долл. Второго приза в 100 долл. был удостоен Джованни Чирауло из Пистои (Италия). Третий приз в 50 долл. достался Дугласу Макдэниелсу из Александрии (шт. Вирджиния).

В заключение я еще раз процитирую высказывание Спаффорда: «Придумать и запустить вирус - это акт не программиста-профессионала, а компьютерного варвара». Пусть те, кто замышляет подобные дела, лучше попробуют свои силы в «Войне в памяти».

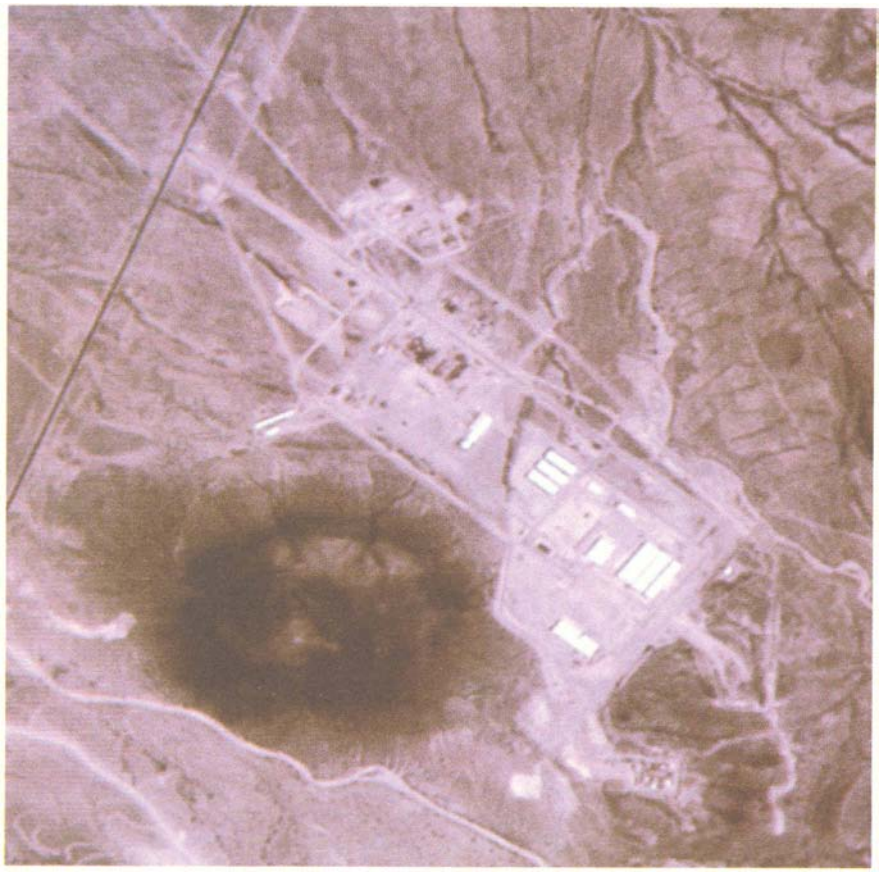
(По всем вопросам, связанным с игрой «Война в памяти» следует обращаться по адресу: 152140, Переславль-Залесский Ярославской обл. Институт программных сетей АН СССР - *Ред.)*

Наука и общество

На пути к договору

В ВЕК ядерных вооружений потенциальную опасность для человечества представляет также и химическое оружие. И тем не менее в наше время его применяют в боевых действиях, что вызывает возмущение у мировой общественности, а картины его использования террористами преследуют людей в ночных кошмарах. И хотя Пентагон намерен и дальше наращивать арсеналы химического оружия, то обстоятельство, что президент Буш поддерживает идею его всеобщего запрета, является хорошим знаком. Каковы же перспективы такого запрета?

Недавние обвинения в том, что торговцы оружием изыскивают пути приобретения бомб с отравляющим веществом (ОВ) нервно-паралитического действия, а химические компании сознательно помогают Ливии в строительстве завода якобы по производству химического оружия, не дают достаточно основания для оптимизма. Тем не менее Джулиан П. Перри Робинсон, главный эксперт ло химическому оружию в Университете графства Суссекс (Великобритания), считает результаты состоявшейся в Париже Международной конференции по запрещению химического оружия обнадеживающими. На ней представители 149 стран подтвердили Женевский протокол 1925 г. о запрещении применения химического оружия и признали продолжающиеся в Женеве переговоры законным форумом для выработки договора о его полном запрещении. Однако заключительное коммюнике Парижской конференции далеко не оправдало надежд некоторых западных стран, в том числе США, ло этому вопросу. Конференция одобрила идею всеобщего запрещения химического оружия, но не поддержала предложения о принятии санкций в отношении тех, кто нарушает указанный Женевский протокол.



ЛИВИЙСКИЙ ПРОМЫШЛЕННЫЙ ОБЪЕКТ в 65 км к югу от Триполи, построенный, по утверждению американских официальных лиц, для производства химического оружия. Ливия отвергает это обвинение. Спутниковая фотография предоставлена фирмой Image Corporation.

И все же, минувший год ознаменовался существенным прогрессом в этой области. На столе женевских переговоров лежит проект договора, нацеленного на уничтожение как заводов по производству химического оружия, так и его накопленных запасов. Договор предусматривает рассекречивание заводов по производству сверхтоксичных отравляющих веществ в исследовательских целях, например зарина и ОВ нервно-паралитического действия, которые смертельны даже в дозах менее 1 мг. Каждая страна сможет иметь не более одного такого завода с ограниченным производством. Договор также предусматривает такую меру, как инспенции по требованию необъявленных предприятий, заподозренных в производстве химического оружия.

По словам сотрудника Гарвардского университета М. Мезелсона, сейчас между делегациями США и Советского Союза на Женевской конференции по разоружению не осталось сколько-нибудь существенных разногласий по данной проблеме. СССР даже принял американское предложение о проведении инспекций по первому требованию «в любое время и в любом месте». Пока договор будет иметь силу, Франция уже не сможет настаивать на сохранении «чрезвычайных запасов» химического оружия. Сохраняющиеся политические разногласия в этой области можно классифицировать по географическому признаку: север - юг. Индия, например, все еще отказывается принять предложение о проведении инспекций по требованию.

Мезелсон и Перри Робертс он считают, что проблема проверки догово-